TEK4030, Mandatory assignment

fridamei

September 2021

1 Exercise 1

a)

$$\frac{X(s)}{U(s)} = \frac{1}{s(1+T_M s)}$$

Finding the poles of the system:

$$U(s) = s(1+T_M s) = 0$$

$$s + T_M s^2 = 0$$

$$s = \frac{-1 \pm \sqrt{1^2 - 4 * 1 * 0}}{2T_M}$$

$$s = \frac{-1-1}{2T_M} \wedge s = \frac{0}{2T_M}$$

$$s = \frac{-1}{T_M} \wedge s = 0$$

The system is marginally stable as the real part of the poles are 0 and negative respectively (given that T_M is positive).

b)

$$H(s) = \frac{X(s)}{U(s)}$$

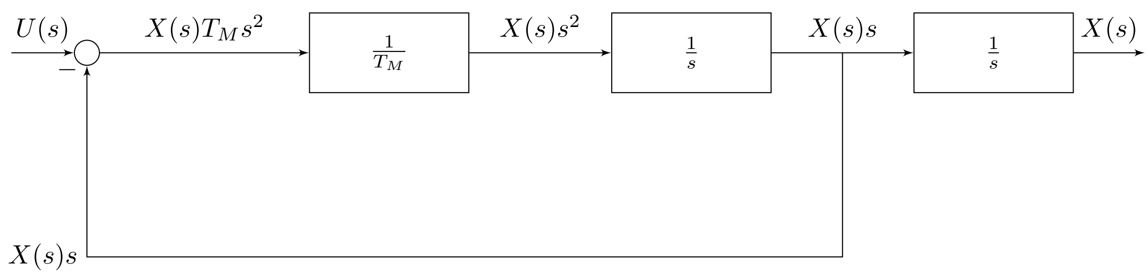
Rewrite equation with respect to input:

$$U(s) = \frac{X(s)}{H(s)}$$

$$U(s) = \frac{X(s)}{\frac{1}{s(1+T_M s)}} = X(s)s + X(s)T_M s^2$$

$$\Rightarrow X(s)T_M s^2 = U(s) - X(s)s$$

This produces the following block diagram:



c)

The system is second order without the controller (because of the s^2 term)
With the controller

$$U(s) = K(1 + T_D s)E(s)$$

where

$$E(s) = R(s) - X(s)$$

This gives:

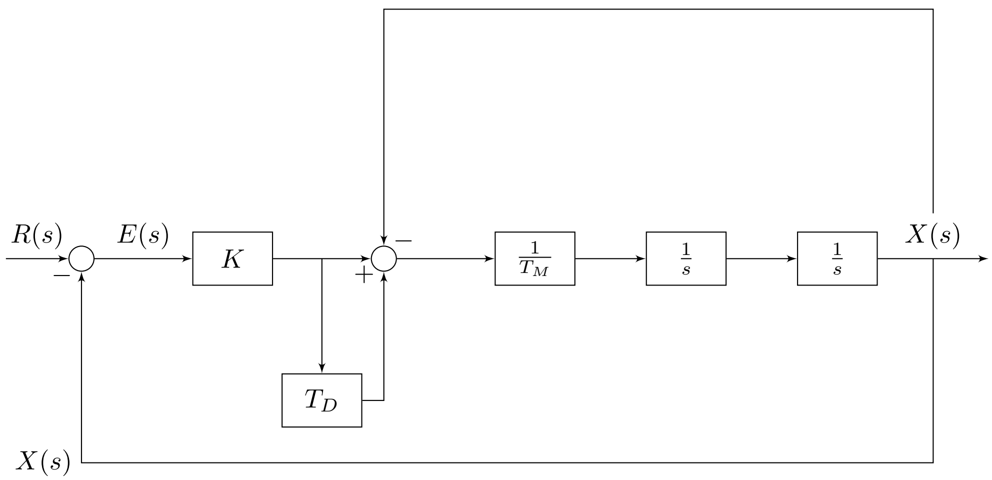
$$\frac{X(s)}{E(s)} = \frac{K(1 + T_D s)}{s(1 + T_M s)}$$

Inserting for E(s) in the above equation will not affect the order, and thus the system with a PD-controller is also second order

d)

$$E(s) = \frac{X(s)}{\frac{K(1+T_D s)}{s(1+T_M s)}} = \frac{X(s)(s(1+T_M s))}{K(1+T_D s)} = \frac{X(s)s + X(s)T_M s^2}{K + T_D s K}$$

$$X(s)T_M s^2 = E(s)K + E(s)T_D s K - X(s)s$$



e)

Transfer function of system with controller:

$$\frac{X(s)}{E(s)} = \frac{K(1+T_D s)}{s(1+T_M s)} \Rightarrow X(s) = \frac{K(1+T_D s) * (R(s) - X(s))}{s(1+T_M s)}$$

$$X(s) = \frac{KR(s) + R(s)KT_D s - KX(s) - X(s)KT_D s}{s(1+T_M s)}$$

$$X(s)s(1+T_M s) + X(s)K + X(s)KT_D s = R(s)(K + KT_D s)$$

$$X(s)(s(1+T_M s) + K + KT_D s) = R(s)(K + KT_D s)$$

$$H(s) = \frac{X(s)}{R(s)} = \frac{K + KT_D s}{s(1+T_M s) + K + KT_D s}$$

$$H(s) = \frac{X(s)}{R(s)} = \frac{K + KT_D s}{T_M s^2 + (1+KT_D)s + K}$$

Zeros:

$$K + KT_D s = 0$$

$$KT_D s = -K$$

$$s = \frac{-1}{T_D}$$

Poles:

$$\frac{K + KT_D s}{T_M s^2 + (1 + KT_D)s + K} = 0$$

$$s = \frac{-(K + KT_D) \pm \sqrt{(K + KT_D)^2 - 4 * T_M K}}{2 * T_D K}$$

Inserting

$$T_M = 2 \wedge T_D = 1$$

Into transfer function and plotting the results

$$H(s) = \frac{X(s)}{R(s)} = \frac{K + Ks}{2s^2 + s(1 + K) + K}$$

polezero.m

```
T_M = 2
T_D = 1
K = linspace(-8, 8, 20)
H = tf([0 T_D*K K], [T_M 1+T_D*K K])

pzplot(H)
grid
```

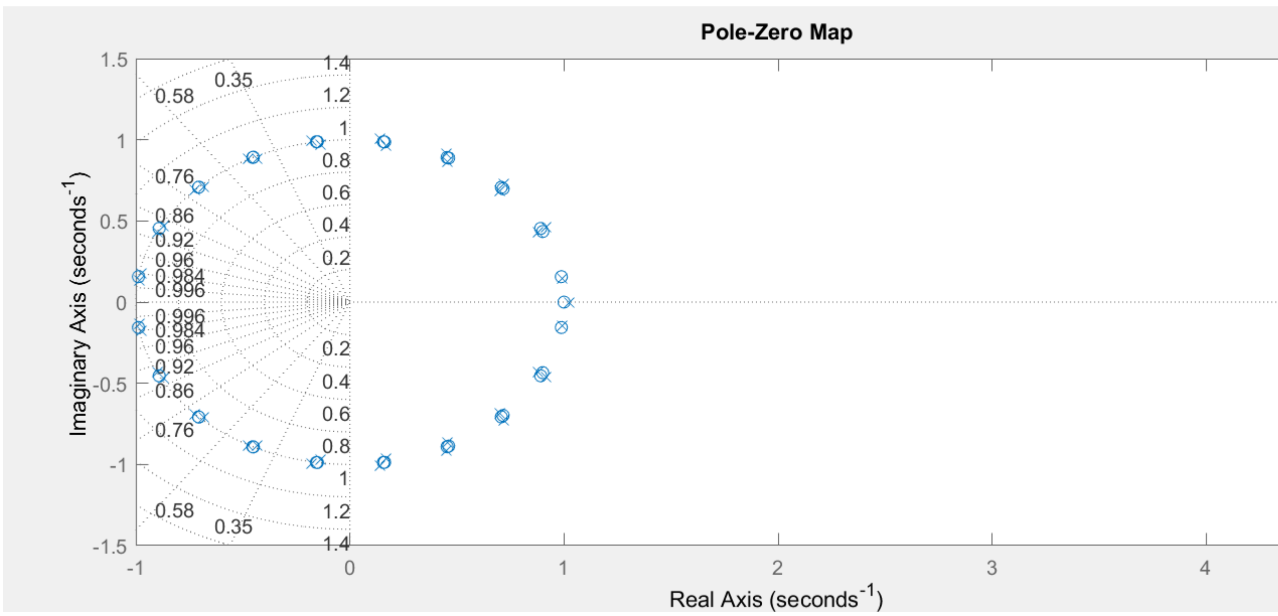


Figure 1: Pole-Zero map of transfer function

2 Exercise 2

a)

Lyapunov stability criteria:

$$I \qquad V(e) > 0 \qquad \forall e \neq 0$$

$$II \qquad V(e) = 0 \qquad e = 0$$
$$III \qquad \dot{V}(e) < 0 \qquad \forall e \neq 0$$
$$IV \qquad V(e) \to \infty \qquad \parallel e \parallel \to \infty$$

System:

$$\dot{x} = -y - x^3$$
$$\dot{y} = x - y^3$$

Candidate function:

$$V(x, y) = x^2 + y^2$$

$$V(x, y) > 0 \qquad \forall e \neq 0$$
$$V(0, 0) = 0$$
$$V(\to \infty, \to \infty) \to \infty$$

For III:

$$\dot{V}(x, y) = 2x\dot{x} + 2y\dot{y} = 0$$

Insert x og y:

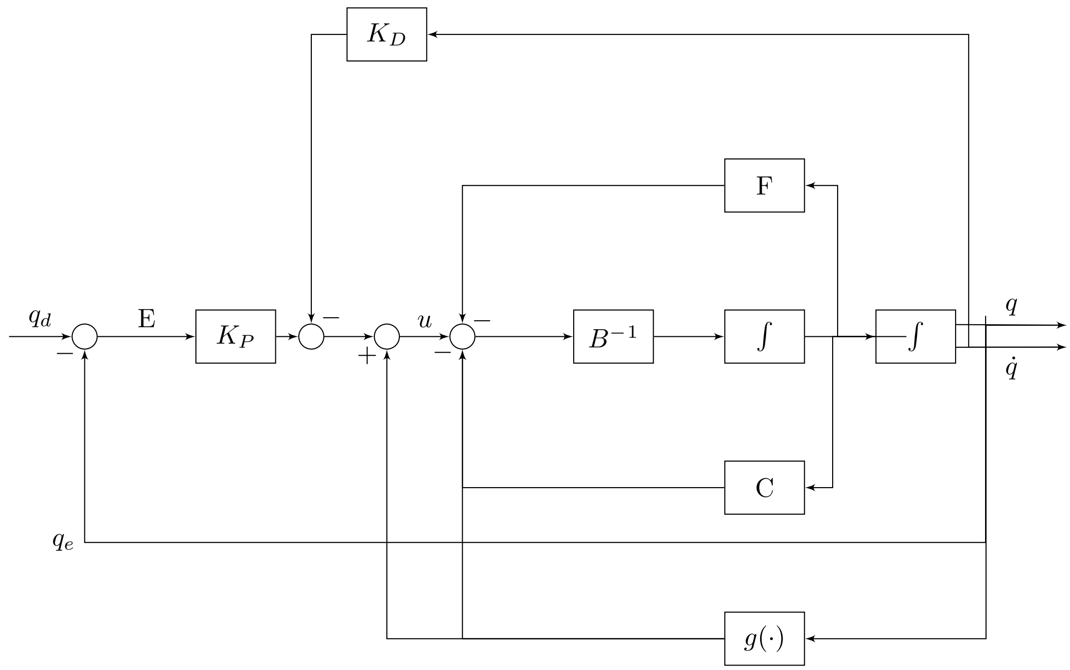
$$2x(-y - x^3) + 2y(x - y^3) = -2xy - 2x^4 + 2xy - 2y^4 = -2x^4 - 2y^4$$

Since $y^4 \wedge \mathrm{x}^4 > 0 \;\forall\; x \wedge y \neq 0,\; then\; \dot{V}(x, y) < 0 \;\forall\; x,\; y \neq 0$

All Lyapunov criteria are met, and therefore the system is globally asymptotically stable

3 Exercise 3

a)



b)

Got satisfactory results with $K_P = 50000$ and $K_D = 6000$, where the steady-state was reached after about 0.6 seconds

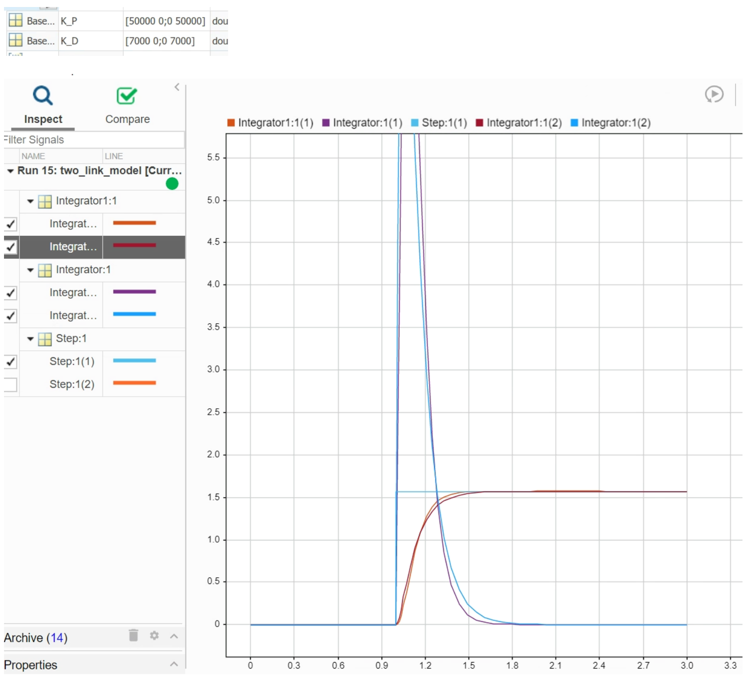


Figure 2: Response of control system

c)

From the book (Robotics: Modelling, Planning and Control by B. Siciliano) we have:

$$(8.57) \qquad u = B(q)y + C(q,\dot{q})\dot{q} + F\dot{q} + g(q)$$

where

$$\ddot{q} = y$$

Further, we have the relationships

$$(8.59) \qquad \ddot{q} + K_D\dot{q} + K_P q = r$$

and

$$(8.60) \qquad r = \ddot{q}_d + K_D\dot{q}_d + K_P q_d$$

Substituting (8.60) into (8.59) gives:

$$\ddot{q} + K_D\dot{q} + K_P q = \ddot{q}_d + K_D\dot{q}_d + K_P q_d$$

$$\Rightarrow \ddot{q} = \ddot{q}_d + K_D\dot{q}_d - K_D\dot{q} + K_P q_d - K_P q$$

$$\Rightarrow \ddot{q} = \ddot{q}_d + K_D(\dot{q}_d - \dot{q}) + K_P(q_d - q)$$

$$\Rightarrow \ddot{q} = \ddot{q}_d + K_D\dot{\tilde{q}} + K_P\tilde{q}$$

Since

$$y = \ddot{q} = \ddot{q}_d + K_D\dot{\tilde{q}} + K_P\tilde{q}$$

Then the control law as a function of q, $\dot{q}$, $\tilde{q}$, $\dot{\tilde{q}}$ and $\ddot{q}_d$ is

$$u = B(q)y + C(q,\dot{q})\dot{q} + F\dot{q} + g(q) = B(q)(\ddot{q}_d + K_D\dot{\tilde{q}} + K_P\tilde{q}) + C(q,\dot{q})\dot{q} + F\dot{q} + g(q)$$

$$u = B(q)(\ddot{q}_d + K_D\dot{\tilde{q}} + K_P\tilde{q}) + C(q,\dot{q})\dot{q} + F\dot{q} + g(q)$$

d)

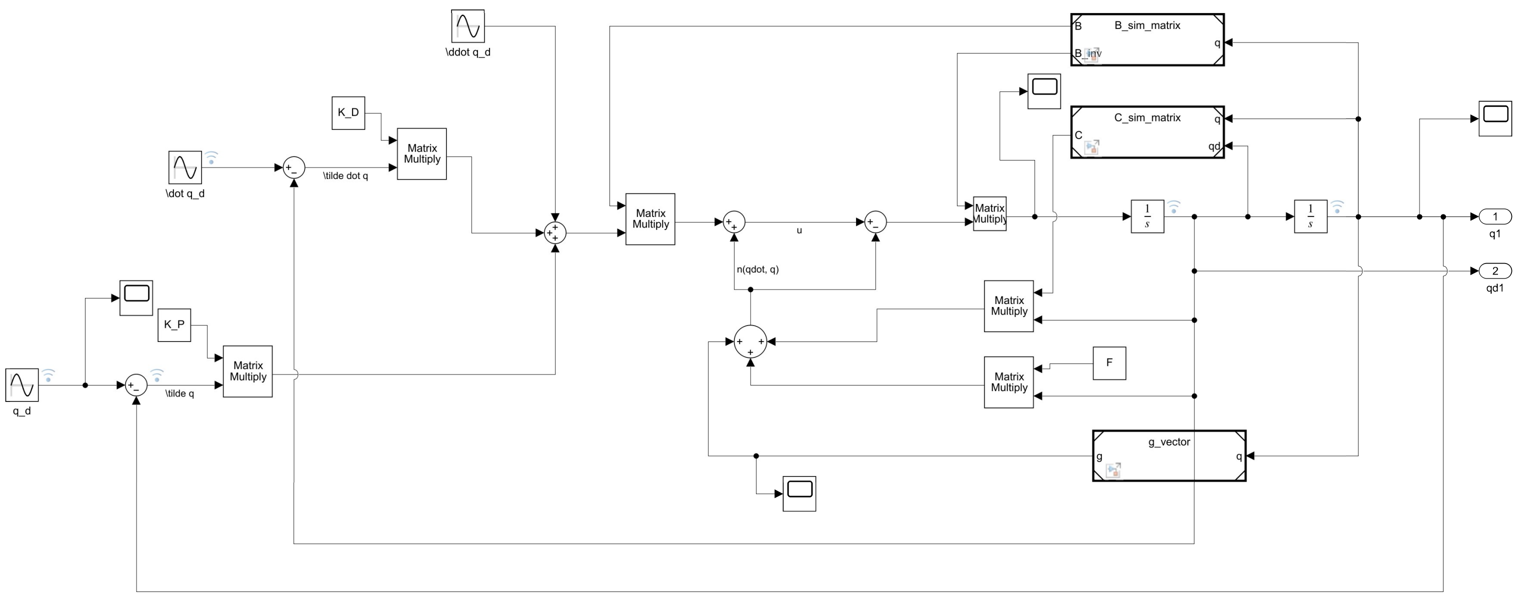


Figure 3: Block diagram of manipulator with controller

e)

Set q_d to $[2\pi f, 2\pi f]$ in a sine wave function generator. $\dot{q}_d$ is the derivative of the desired joint position, and thus is set to frequency $[2\pi f, 2\pi f]$ with amplitude $2\pi f$. Since the wave generator produces sine waves, the phase is set to $\frac{\pi}{2}$ to produce the desired cosine waves. $\ddot{q}_d$ is the derivative of $\dot{q}$ and therefore has frequency $[2\pi f, 2\pi f]$ with amplitude $-4\pi^2 f^2$. The phase is 0 as the desired wave is a sine wave.

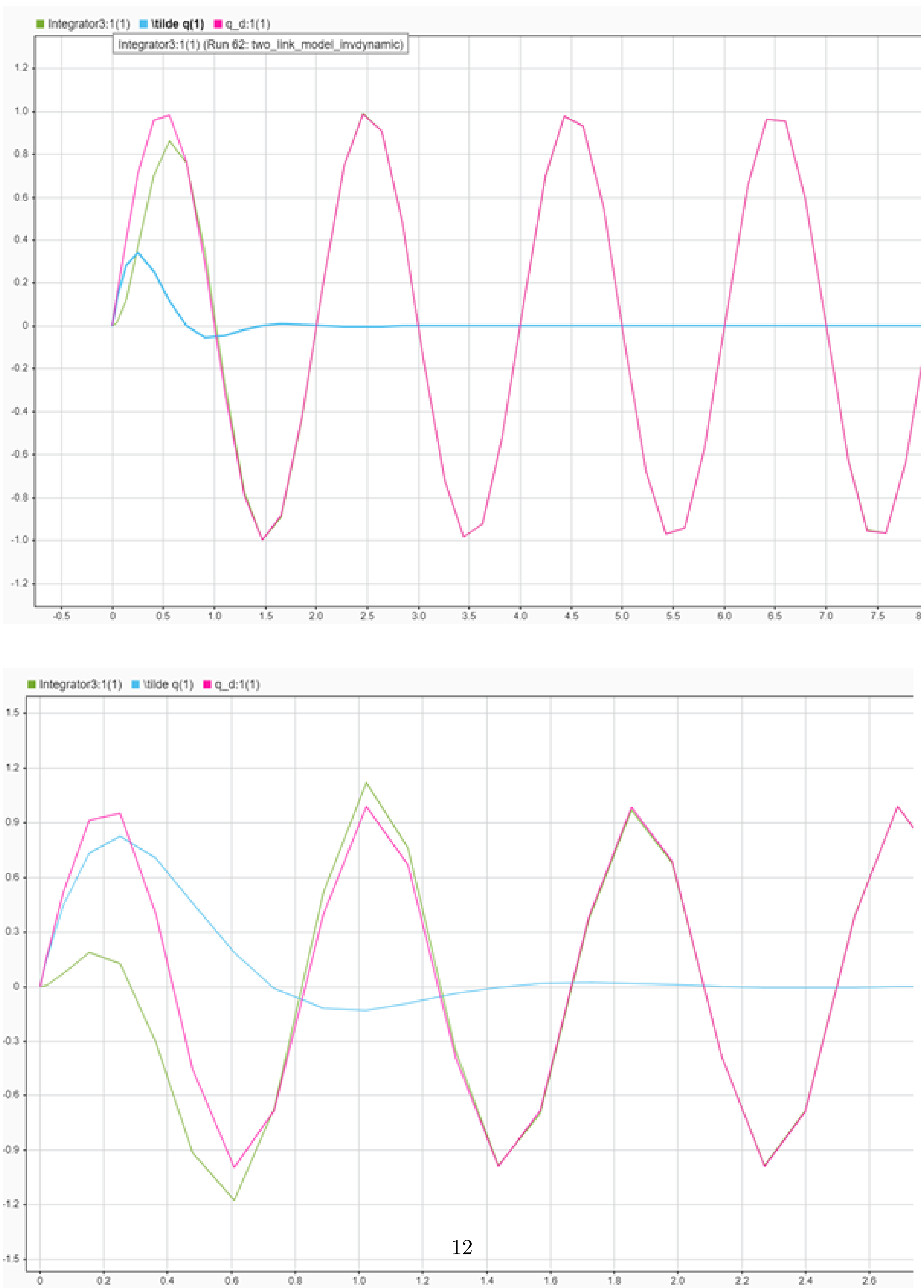


Figure 4: Plots with f = 0.5 (on top) and f = 1.2. q_d is shown as a red line, q as a green line and the error between the two as a blue line

The error ($\tilde{q}_d$) is shown in blue in the above plots. It oscillates some before settling at 0. Seems like the system is properly damped when f = 0.5, but when it is increased to 1.2, the system overshoots and as stated the error has some oscillation and is therefore under-damped

f)
A critically damped system is found when the damping ratio ζ is equal to 1. $K_P = diag\{\omega_{n1}^2, \omega_{n2}^2\}$ and $K_D = diag\{2\zeta_1\omega_{n1}^2, 2\zeta_2\omega_{n2}^2\}$.

With $\zeta = 1$ and $\omega = 5$, this gives

$$K_P = diag\{25, 25\}$$

$$K_D = diag\{10, 10\}$$

Simulated with f = 2 produces the following plot

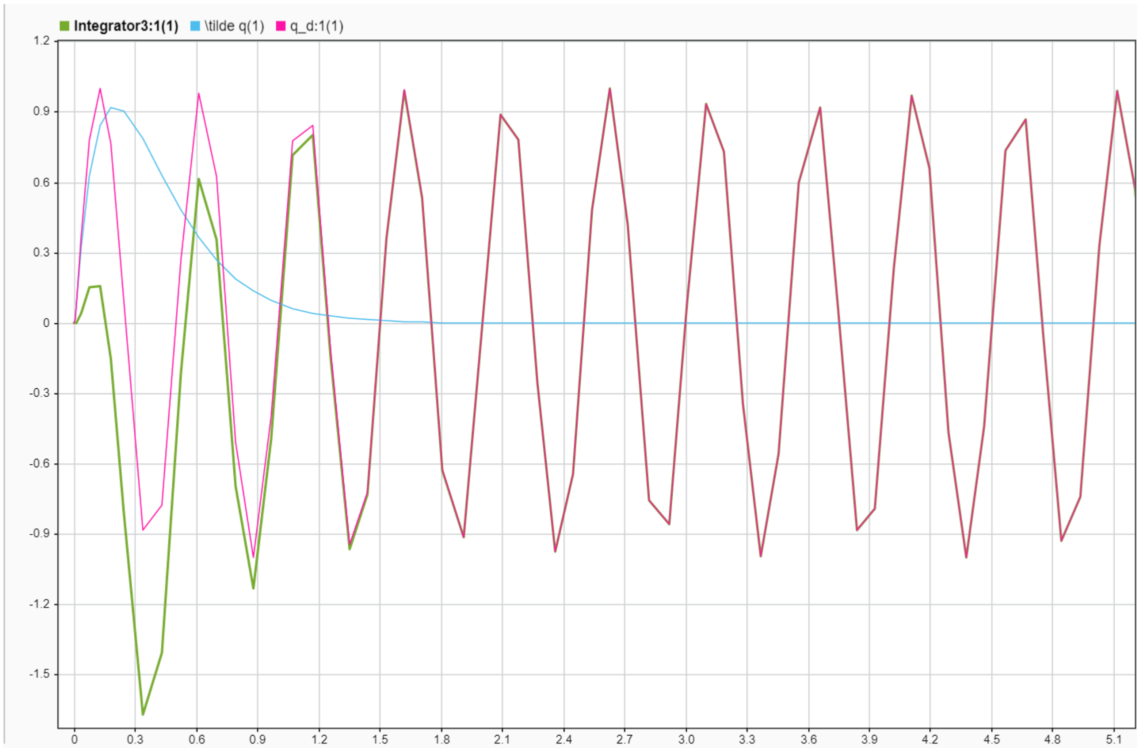


Figure 5: Critically damped with f = 2
The error quickly settles at 0 without oscillations

g)

With $\hat{B} = 0.9B$ and $\hat{n} = 0.95n$ I got the following results (not sure if I implemented the error correctly though)

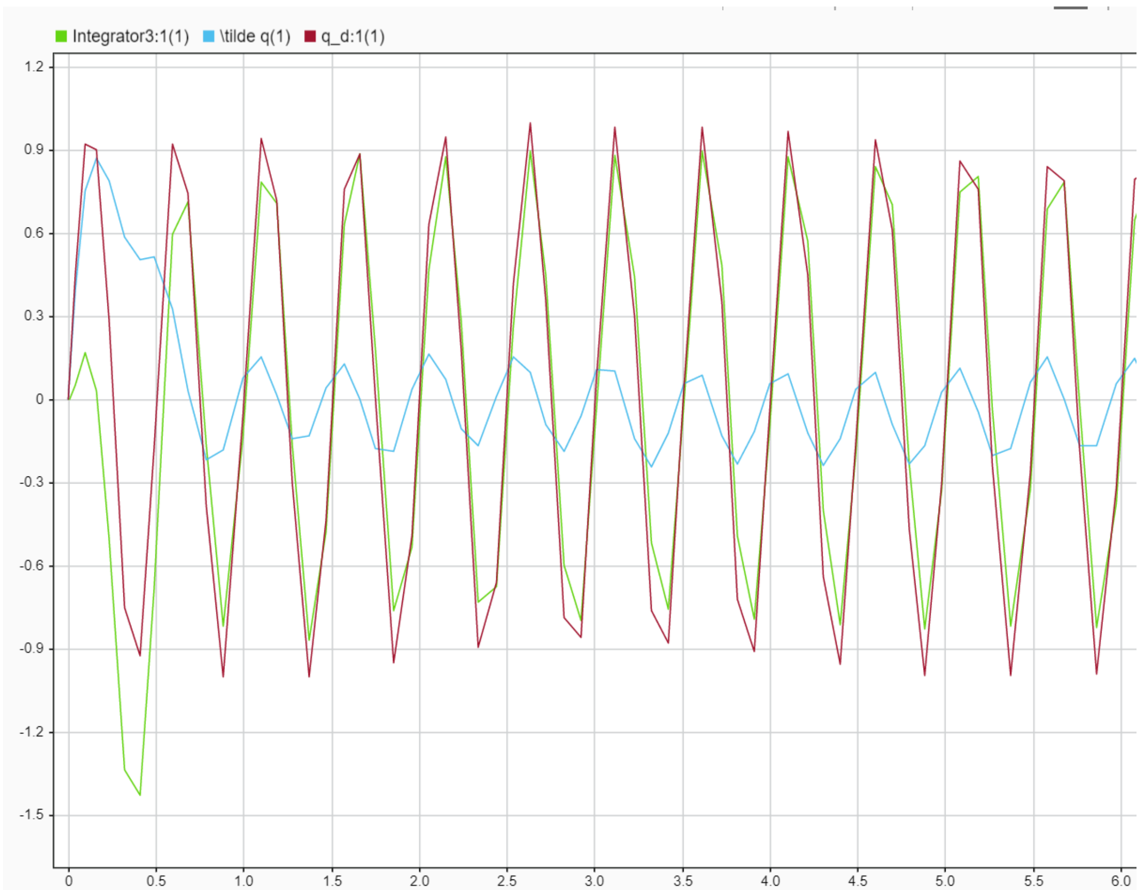


Figure 6: With error terms $\hat{B} = 0.9B$ and $\hat{n} = 0.95n$
The error never settles and the output never quite reaches the set point.

4 Exercise 4

a)

Expanding the nodes gave the following controller.cpp and simulator.cpp code (the files are also attached):

Controller code:

```
#include <ros/ros.h>
#include <sensor_msgs/JointState.h>

#include <std_msgs/Float64MultiArray.h>


#include <planar_robot_simulator/planar_robot_2dof.h>

using namespace PlanarRobotSimulator;
using namespace PlanarRobotSimulator::Parameters;


ros::Publisher* command_pub_global = NULL;

void jointStateCallback(const sensor_msgs::JointState::ConstPtr& msg)
{

  Eigen::Vector2d q_m(msg->position[0], msg->position[1]);
  Eigen::Vector2d qd_m(msg->velocity[0], msg->velocity[1]);


  Eigen::Vector2d q_m_set(0.5*M_PI*k_r_1, -0.25*M_PI*k_r_2);

  Eigen::Vector2d K_p(1.0, 1.0);
  Eigen::Vector2d u = K_p.cwiseProduct(q_m_set-q_m);

  std_msgs::Float64MultiArray output;
  output.data.push_back(u(0));
  output.data.push_back(u(1));

  command_pub_global->publish(output);
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "controller");
  ros::NodeHandle nh;


  ros::Publisher command_pub = nh.advertise<std_msgs::Float64MultiArray>("joint_command", 1000);

  command_pub_global = &command_pub;

  ros::Subscriber sub = nh.subscribe("joint_state", 1000, jointStateCallback);

  ros::spin();

  return 0;
}
```

Simulator code:

```
#include <ros/ros.h>
#include <planar_robot_simulator/planar_robot_2dof.h>
#include <sensor_msgs/JointState.h>
```

```

#include <std_msgs/Float64MultiArray.h>

Eigen::Vector2d u;

void jointCommandCallback(const std_msgs::Float64MultiArray::
    ConstPtr& msg)
{

  Eigen::Vector2d u_local(msg->data[0], msg->data[1]);
  u = u_local;

}


int main(int argc, char **argv)
{
  ros::init(argc, argv, "simulator");
  ros::NodeHandle nh;


/* This is the object that simulates a pendulum */
  PlanarRobotSimulator::PlanarRobot2DOF sim;

  ros::Rate loop_rate(100);

    /* Allocate the message */
  sensor_msgs::JointState msg;
  msg.name.push_back("joint_1");
  msg.name.push_back("joint_2");
  msg.position.push_back(0.0);
  msg.position.push_back(0.0);
  msg.velocity.push_back(0.0);
  msg.velocity.push_back(0.0);

  /* Allocate publishers */
  ros::Publisher joint_state_pub = nh.advertise<sensor_msgs::
    JointState>("joint_state", 1000);

  /* Allcate subscibers */
  ros::Subscriber sub = nh.subscribe("joint_command", 1000,
    jointCommandCallback);


  while (ros::ok())
  {
    ros::spinOnce();

    /* This performs one step in the simulation */
    double dt = 1.0/100.0;

    sim.step(dt, u);

    /* Set the joint state */
    msg.header.stamp = ros::Time::now();

   Eigen::Vector2d q_m = sim.getMotorPosition();
```

```
    msg.position[0] = q_m(0);
    msg.position[1] = q_m(1);

    Eigen::Vector2d qd_m = sim.getMotorVelocity();
    msg.velocity[0] = qd_m(0);
    msg.velocity[1] = qd_m(1);

     joint_state_pub.publish(msg);

     /* This creates a window that shows the pendulum */
     sim.draw();
     loop_rate.sleep();
  }

  return 0;
}
```

b)

$$(8.11) \qquad \tau = K_r K_t R_a^{-1}(G_v v_c - K_v K_r \dot{q})$$

$$\tau = K_r K_t R_a^{-1} G_v v_c - K_r K_t R_a^{-1} K_v K_r \dot{q}$$

Substituting control input

$$u = K_t R_a^{-1} G_v v_c$$

into (8.11):

$$\tau = K_r u - K_r K_t R_a^{-1} K_v K_r \dot{q}$$

We have that

$$\tau_m = K_r^{-1} \tau$$

Inserted into above equation yields

$$K_r^{-1}\tau = K_r^{-1} K_r u - K_r^{-1} K_r K_t R_a^{-1} K_v K_r \dot{q}$$

$$\rightarrow \tau_m = u - K_t R_a^{-1} K_v K_r \dot{q}$$

$$\boxed{\dot{q}_m = K_r \dot{q}}$$

$$\rightarrow \tau_m = u - K_t R_a^{-1} K_v \dot{q}_m$$

$$(8.18) \qquad K^{-1}B(q)K^{-1}\ddot{q}_m + K^{-1}C(q,\dot{q})K^{-1}\dot{q}_m + K^{-1}F_v K^{-1}\dot{q}_m + K^{-1}g(q) = \tau_m$$

Substituting τ_m in (8.18) with the above expression yields

$$K^{-1}B(q)K^{-1}\ddot{q}_m + K^{-1}C(q,\dot{q})K^{-1}\dot{q}_m + K^{-1}F_v K^{-1}\dot{q}_m + K^{-1}g(q) = u - K_t R_a^{-1} K_v K_r \dot{q}$$

$$\rightarrow K^{-1}B(q)K^{-1}\ddot{q}_m + K^{-1}C(q,\dot{q})K^{-1}\dot{q}_m + K^{-1}F_v K^{-1}\dot{q}_m + K_t R_a^{-1} K_v \dot{q}_m + K^{-1}g(q) = u$$

Set all configuration independent terms containing $\dot{q}_m$ as variable F_m

$$F_m = K^{-1}F_v K^{-1} + K_t R_a^{-1} K_v$$

which yields

$$u = K^{-1}B(q)K^{-1}\ddot{q}_m + K^{-1}C(q,\dot{q})K^{-1}\dot{q}_m + F_m \dot{q}_m + K^{-1}g(q)$$

c)

Setting

$$B(q) = \bar{B} + \Delta B(q)$$

where $\bar{B}$ is a diagonal matrix with elements representing average inertia at each joint and $\Delta B(q)$ is configuration dependent.
The configuration dependent terms (nonlinear coupled):

$$d = K^{-1}\Delta B(q)K^{-1}\ddot{q}_m + K^{-1}C(q,\dot{q})K^{-1}\dot{q}_m + K^{-1}g(q)$$

The linear decoupled terms:

$$K^{-1}\bar{B}(q)K^{-1}\ddot{q}_m + F_m\dot{q}_m$$

Combined:

$$u = K^{-1}\bar{B}(q)K^{-1}\ddot{q}_m + F_m\dot{q}_m + d$$